ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**Отчет**

**«АНИМАЦИЯ ТОЧКИ»**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА**

**И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»**

**ВАРИАНТ ЗАДАНИЯ №15**

Выполнила студентка группы М8О-206Б-23

Осташевская А.И. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                                                                          подпись, дата

Проверил и принял

Волков А.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                                                                         подпись, дата

**Москва 2024**

**Вариант № 15**

**Задание:**

Построить заданную траекторию и анимацию движения точки, а также отобразить стрелки радиус-вектора, скорости и ускорения. Построить радиус кривизны траектории.

**Закон движения точки:**

**Текст программы**

Из полярной СК надо в прямоугольную.

RX,RY это координаты в ДСК .

=

phi это

Производная по времени нужна чтобы более точно построить график.Найти асимптоты,критические точки, экстремумы,радиус-вектор.Более точное описание функции ,чтобы её построить.Разложить ускорения на тангенциальное, нормальное.Так же напишу, что для подсчёта я вызвала функции подсчёта дифференциала библиотеку numpy.

Vphi это дифференциал (производная по времени).

import math

import sympy as s

import matplotlib.pyplot as plt

import numpy as np

from matplotlib.animation import FuncAnimation

# Функция для отрисовки стрелок векторов

def Vect\_arrow(X, Y, ValX, ValY):

a = 0.2

b = 0.3

Arx = np.array([-b, 0, -b])

Ary = np.array([a, 0, -a])

alp = math.atan2(ValY, ValX)

RotArx = Arx \* np.cos(alp) - Ary \* np.sin(alp)

RotAry = Arx \* np.sin(alp) + Ary \* np.cos(alp)

Arx = X + ValX + RotArx

Ary = Y + ValY + RotAry

return Arx, Ary

# Переменная времени

t = s.Symbol('t')

# Заданные траектории

x = s.cos(1.8 \* t + 0.2 \* (s.cos(12 \* t)\*2)) \* (2 + s.sin(12 \* t))

y = s.sin(1.8 \* t + 0.2 \* (s.cos(12 \* t)\*2)) \* (2 + s.sin(12 \* t))

# Скорость

Vx = s.diff(x)

Vy = s.diff(y)

# Ускорение

Ax = s.diff(Vx)

Ay = s.diff(Vy)

# Радиус кривизны

rho = (Vx\*\*2 + Vy\*\*2) / s.sqrt((Ax\*\*2 + Ay\*\*2) - (s.diff(s.sqrt(Vx\*\*2 + Vy\*\*2), t))\*\*2)

# Преобразование выражений в функции для численных вычислений

F\_x = s.lambdify(t, x, 'numpy')

F\_y = s.lambdify(t, y, 'numpy')

F\_Vx = s.lambdify(t, Vx, 'numpy')

F\_Vy = s.lambdify(t, Vy, 'numpy')

F\_Ax = s.lambdify(t, Ax, 'numpy')

F\_Ay = s.lambdify(t, Ay, 'numpy')

F\_rho = s.lambdify(t, rho, 'numpy')

# Параметры анимации

step = 700

#T = np.linspace(1.5, 2.2, step) # временной интервал

T = np.linspace(0, 20, step) # временной интервал

# Вычисление значений

X = F\_x(T)

Y = F\_y(T)

VX = F\_Vx(T)

VY = F\_Vy(T)

AX = F\_Ax(T)

AY = F\_Ay(T)

Rho = F\_rho(T)

# Создание окна

fig = plt.figure("lab1")

axis = fig.add\_subplot(1, 1, 1)

axis.axis('equal')

axis.set(xlim=[-10, 10], ylim=[-10, 10])

# Отрисовка траектории

axis.plot(X, Y, label='Траектория')

axis.legend()

# Начальное положение объектов

Pnt = axis.plot([X[0]], [Y[0]], marker='o')[0]

Vp = axis.plot([X[0], X[0] + VX[0]], [Y[0], Y[0] + VY[0]], 'red', label='Вектор скорости')[0]

Ap = axis.plot([X[0], X[0] + AX[0]], [Y[0], Y[0] + AY[0]], 'blue', label='Вектор ускорения')[0]

RhoX = X[0] + VY[0] \* Rho[0] / math.sqrt(VX[0]\*\*2 + VY[0]\*\*2)

RhoY = Y[0] - VX[0] \* Rho[0] / math.sqrt(VX[0]\*\*2 + VY[0]\*\*2)

Rhop = axis.plot([X[0], RhoX], [Y[0], RhoY], 'black', label='Радиус кривизны')[0]

RLine = axis.plot([0, X[0]], [0, Y[0]], 'orange', label='Радиус-вектор')[0]

# Отрисовка стрелок

RAx1, RAy1 = Vect\_arrow(X[0], Y[0], VX[0], VY[0])

Varrow = axis.plot(RAx1, RAy1, 'red')[0]

RAx2, RAy2 = Vect\_arrow(X[0], Y[0], AX[0], AY[0])

Aarrow = axis.plot(RAx2, RAy2, 'blue')[0]

RAx3, RAy3 = Vect\_arrow(X[0], Y[0], RhoX - X[0], RhoY - Y[0])

RHarrow = axis.plot(RAx3, RAy3, 'black')[0]

RAx4, RAy4 = Vect\_arrow(0, 0, X[0], Y[0])

Rarrow = axis.plot(RAx4, RAy4, 'orange')[0]

# Анимация

def anim(i):

RhoX = X[i] + VY[i] \* Rho[i] / math.sqrt(VX[i]\*\*2 + VY[i]\*\*2)

RhoY = Y[i] - VX[i] \* Rho[i] / math.sqrt(VX[i]\*\*2 + VY[i]\*\*2)

# Обновление положения

Pnt.set\_data([X[i]], [Y[i]])

Vp.set\_data([X[i], X[i] + VX[i]], [Y[i], Y[i] + VY[i]])

Ap.set\_data([X[i], X[i] + AX[i]], [Y[i], Y[i] + AY[i]])

Rhop.set\_data([X[i], RhoX], [Y[i], RhoY])

RLine.set\_data([0, X[i]], [0, Y[i]])

# Обновление стрелок

RAx1, RAy1 = Vect\_arrow(X[i], Y[i], VX[i], VY[i])

Varrow.set\_data(RAx1, RAy1)

RAx2, RAy2 = Vect\_arrow(X[i], Y[i], AX[i], AY[i])

Aarrow.set\_data(RAx2, RAy2)

RAx3, RAy3 = Vect\_arrow(X[i], Y[i], RhoX - X[i], RhoY - Y[i])

RHarrow.set\_data(RAx3, RAy3)

RAx4, RAy4 = Vect\_arrow(0, 0, X[i], Y[i])

Rarrow.set\_data(RAx4, RAy4)

# Запуск анимации

an = FuncAnimation(fig, anim, frames=step, interval=1)

plt.show()

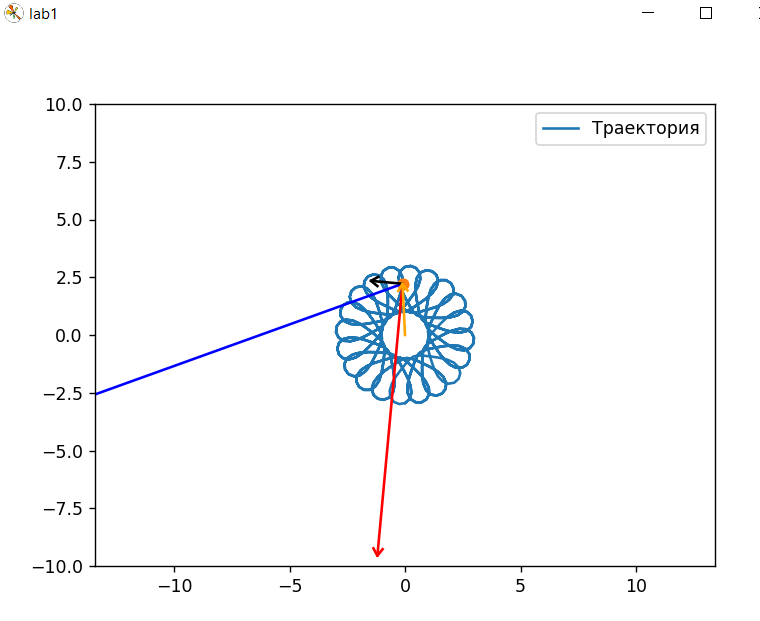
См.след страницу

**Результат работы программы:**

 **Синяя стрелка** показывает, как изменяется скорость объекта (ускорение).

 **Желтая стрелка** указывает положение объекта относительно начала координат.(0, 0)

 **Черная стрелка** характеризует "закругленность" траектории в текущей точке (радиус кривизны).



|  |
| --- |
|  |
|  | |

**Вывод:**

Программа вычислила все необходимые тесты. Петля строится от времени это не означает последовательное прорисовка витка . Прорисовка зависит от времени .и в разные моменты времени точка отображается на координаты x и y причем значения скачут как угодно и именно поэтому они изменяются в данном случае не линейно (как например y=sin(t) или y=t ).

r(t) является циклоидой. Радиус кривизны меняется вдоль траектории. Это указывает на то, что объект движется по участкам с разной кривизной. Так же вот такой вывод по кривизне имеется:

* Участки с резким поворотом имеют малый радиус кривизны.
* Прямые или более плавные участки имеют большой радиус кривизны.

Ссылка на гитхаб с данной лабораторной работой:

https://github.com/Garnet-t/Theorethical-mechanics/tree/main/lab1